

### Claims

- 1 1. A method of parallel processing comprising:
- 2 providing a first thread which represents an independent flow of control managed
- 3 by a program structure, said first thread having two states, a first state
- 4 processing work for the program structure and a second state undispatched
- 5 awaiting work to process;
- 6 providing a second thread which represents an independent flow of control managed
- 7 by a program structure separate from the first thread;
- 8 using the second thread to prepare work for the first thread to process;
- 9 placing the work prepared by the second thread in a queue for processing by the
- 10 first thread;
- 11 if the first thread is awaiting work to process when the work prepared by the
- 12 second thread is placed in the queue, dispatching the first thread and using it to
- 13 process the work in the queue;
- 14 if the first thread is processing other work when the work prepared by the second
- 15 thread is placed in the queue, using the first thread to complete processing of
- 16 the other work, access the work in the queue, and then process the work in the
- 17 queue.
- 1 2. The method of claim 1 wherein the second thread continues to place additional
- 2 work in the queue, and the first thread sequentially processes the additional work in the
- 3 queue as it completes processing prior work.
- 1 3. The method of claim 1 wherein the second thread marks the work placed in the
- 2 first thread queue as not complete.
- 1 4. The method of claim 1 wherein if the first thread is processing other work when
- 2 the work prepared by the second thread is placed in the queue, and when the first

3 thread completes processing of the work in the queue, using the first thread to mark the  
4 completed work as complete, wherein subsequent work from the second thread is made  
5 to wait until the previous work in the first thread is marked complete.

1 5. The method of claim 1 wherein the first thread is reused to process other work.

2 6. The method of claim 4 wherein the program structure destroys the first thread  
after it completes a desired amount of work.

3 7. A method of parallel processing comprising:  
4 providing a first thread which represents an independent flow of control managed  
5 by a program structure, said first thread having two states, a first state  
6 processing work for the program structure and a second state undispatched  
7 awaiting work to process;  
8 providing a second thread which represents an independent flow of control managed  
9 by a program structure separate from the first thread;  
10 using the second thread to prepare work for the first thread to process;  
11 placing the work prepared by the second thread in a queue for processing by the  
12 first thread, the work placed in the first thread queue being marked as not  
13 complete;  
14 if the first thread is awaiting work to process when the work prepared by the  
15 second thread is placed in the queue, dispatching the first thread and using it to  
16 process the work in the queue;  
17 if the first thread is processing other work when the work prepared by the second  
18 thread is placed in the queue, using the first thread to complete processing of  
19 the other work, access the work in the queue, and then process the work in the  
20 queue;  
21 using the second thread to place additional work in the queue; and

22 using the first thread to sequentially process the additional work in the queue as it  
23 completes processing prior work.

1 8. The method of claim 7 wherein if the first thread is processing other work when  
2 the work prepared by the second thread is placed in the queue, and when the first  
3 thread completes processing of the work in the queue, using the first thread to mark the  
4 completed work as complete, wherein subsequent work from the second thread is made  
5 to wait until the previous work in the first thread is marked complete.

1 9. The method of claim 7 the first thread is reused to process other work.

1 10. The method of claim 8 wherein the program structure destroys the first thread  
2 after it completes a desired amount of work.

1 11. A program storage device readable by a machine, tangibly embodying a  
2 program of instructions executable by the machine to perform method steps of parallel  
3 processing using i) a first thread which represents an independent flow of control  
4 managed by a program structure, said first thread having two states, a first state  
5 processing work for the program structure and a second state undispached awaiting  
6 work to process, and ii) a second thread which represents an independent flow of  
7 control managed by a program structure separate from the first thread, said method  
8 steps comprising:

9 using the second thread to prepare work for the first thread to process;  
10 placing the work prepared by the second thread in a queue for processing by the  
11 first thread;  
12 if the first thread is awaiting work to process when the work prepared by the  
13 second thread is placed in the queue, dispatching the first thread and using it to  
14 process the work in the queue;

15 if the first thread is processing other work when the work prepared by the second  
16 thread is placed in the queue, using the first thread to complete processing of  
17 the other work, access the work in the queue, and then process the work in the  
18 queue.

Sub A1 12. The program storage device of claim 11 wherein, in the method steps, the  
1 second thread continues to place additional work in the queue, and the first thread  
2 sequentially processes the additional work in the queue as it completes processing prior  
3 work.  
4

1 13. The program storage device of claim 11 wherein, in the method steps, the  
2 second thread marks the work placed in the first thread queue as not complete.

1 14. The program storage device of claim 11 wherein, in the method steps, if the  
2 first thread is processing other work when the work prepared by the second thread is  
3 placed in the queue, and when the first thread completes processing of the work in the  
4 queue, using the first thread to mark the completed work as complete, wherein  
5 subsequent work from the second thread is made to wait until the previous work in the  
6 first thread is marked complete.

1 15. The program storage device of claim 11 wherein, in the method steps, the first  
2 thread is reused to process other work.

1 16. The program storage device of claim 14 wherein, in the method steps, the  
2 program structure destroys the first thread after it completes a desired amount of work.